

refined search space. Whichever of the other software projects in the refined search space has the closest distance to the modified feature vector can be the closest match to Tensorflow. This example is simply illustrative and a similar process can be applied to software projects other than Tensorflow. Similarly, other examples can involve a more components, fewer components, or a different arrangement of the components shown in FIG. 1.

[0027] FIG. 2 is a block diagram of another example of a system 200 for automatically building a database 116 of software features for software projects according to some aspects. The system 200 includes a processor 302 communicatively coupled with a memory device 204. The processor 302 and the memory device 204 may or may not be part of a computing device, such as computing device 102.

[0028] The processor 302 can include one processor or multiple processors. Non-limiting examples of the processor 302 include a Field-Programmable Gate Array (FPGA), an application-specific integrated circuit (ASIC), a microprocessor, etc. The processor 202 can execute instructions 206 stored in the memory device 204 to perform operations. In some examples, the instructions 206 can include processor-specific instructions generated by a compiler or an interpreter from code written in any suitable computer-programming language, such as C, C++, C #, etc.

[0029] The memory device 204 can include one memory device or multiple memory devices. The memory device 204 can be non-volatile and may include any type of memory device that retains stored information when powered off. Non-limiting examples of the memory device 204 include electrically erasable and programmable read-only memory (EEPROM), flash memory, or any other type of non-volatile memory. In some examples, at least some of the memory device can include a medium from which the processor 202 can read instructions 206. A non-transitory computer-readable medium can include electronic, optical, magnetic, or other storage devices capable of providing the processor 302 with computer-readable instructions 206 or other program code. Non-limiting examples of a non-transitory computer-readable medium include magnetic disk(s), memory chip(s), ROM, random-access memory (RAM), an ASIC, a configured processor, optical storage, or any other medium from which a computer processor can read the instructions 206.

[0030] In some examples, the processor 202 can analyze descriptive information 104 about a software project 122 to determine software features 110 of the software project 122. The processor 202 can then generate a feature vector 114 for the software project 122 based on the software features 110 of the software project 122. The feature vector 114 can be a vector of elements 208 in which each element is a numerical value indicating whether a particular software feature corresponding to the element is among the software features 110 of the software project 122 as determined from the descriptive information 104. The processor 202 can then store the feature vector 114 in a database 116 having a group of feature vectors for a group of software projects. The group of feature vectors can be searchable in response to search queries 118.

[0031] In some examples, the processor 202 can perform one or more of the steps shown in FIG. 3 according to some aspects. In other examples, the processor 202 can implement more steps, fewer steps, different steps, or a different order

of the steps depicted in FIG. 3. The steps of FIG. 3 are described below with reference to components discussed above.

[0032] In block 302, a processor 202 analyzes descriptive information 104 about a software project 122 to determine software features 110 of the software project 122. For example, the processor 202 can obtain the descriptive information 104 from one or more sources. The processor 202 can then apply a count technique and/or a machine-learning model to the descriptive information 104 to determine software features 110 of the software project 122.

[0033] In block 304, the processor 202 generates a feature vector 114 for the software project 122 based on the software features 110 of the software project 122. The feature vector 114 can be a data structure (e.g., vector) of elements 208 in which each element is a numerical value indicating whether a particular software feature corresponding to the element is among the software features 110 of the software project 122 as determined from the descriptive information 104. The processor 202 can generate the feature vector 114 by setting element values (e.g., bit values) in the feature vector 114 based on the software features 110 of the software project 122.

[0034] In block 306, the processor 202 stores the feature vector 114 in a database 116 having a plurality of feature vectors for a plurality of software projects. The plurality of feature vectors can be searchable in response to search queries 118.

[0035] In some examples, the blocks 302-306 can be repeated for as many software projects as desired to automatically construct a database 116 of feature vectors, which can be easily searched and compared. This may enable developers to quickly and accurately identify software applications that are compatible with their specific computing environments, that satisfy particular computing criteria, and/or that can serve as suitable replacements for existing software applications in their computing environments.

[0036] The foregoing description of certain examples, including illustrated examples, has been presented only for the purpose of illustration and description and is not intended to be exhaustive or to limit the disclosure to the precise forms disclosed. Numerous modifications, adaptations, and uses thereof will be apparent to those skilled in the art without departing from the scope of the disclosure. Some examples can be combined with other examples to yield further examples.

1. A system comprising:
 - a processor; and
 - a memory device includes instructions that are executable by the processor for causing the processor to:
 - analyze descriptive information about a software project to determine software features of the software project, the software features being functional characteristics of the software project;
 - generate a feature vector for the software project based on the software features of the software project, the feature vector being a data structure of elements in which each element is a numerical value indicating whether a particular software feature corresponding to the element is among the software features of the software project as determined from the descriptive information; and
 - store the feature vector in a database having a plurality of feature vectors for a plurality of software projects,